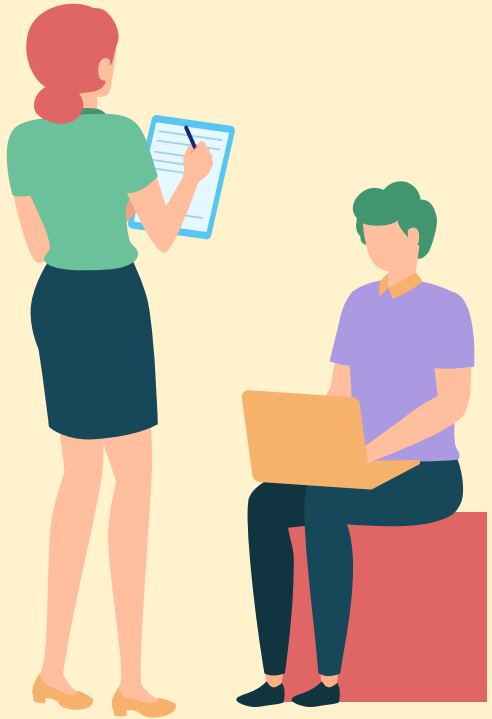
Marist Banner Tracker

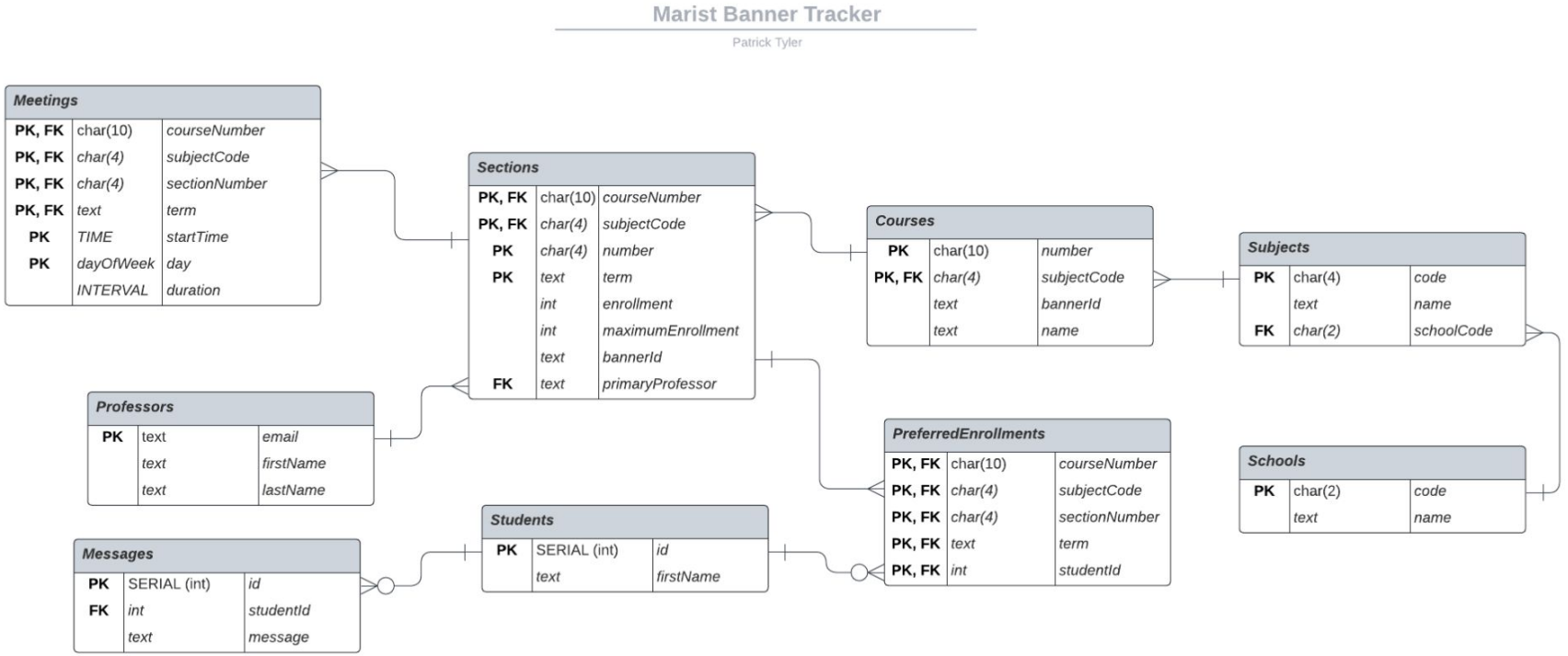
Patrick Tyler
CMPT 308
May 2024

Executive Summary



This is a system engineered for Marist students to track their sections throughout the registration process. Students are annoyed with uncertainty during registration. This system aims to provide them with live and accurate information and programmatically notifying students on changes to sections they plan on registering for. This system takes all class data for a term at multiple points in the registration process updating its existing data accordingly.

Entity-Relationship Diagram



Tables

```
CREATE TABLE Professors (
    -- sort of bad practice to use email because it might change but it
    --     makes the data entry easier
    email text PRIMARY KEY,
    firstName text,
    lastName text
);
```

- Strong entity to represent Marist Professors
- Email is chosen as primary key even though it is generally discouraged because the Marist Banner website only exposes email as identifying professor information

Functional Dependency: **email** -> firstName, lastName

Professors - Sample Data

email	firstname	lastname
Amanda.Damiano@marist.edu	Amanda	Damiano
Heather.Hallenbeck@marist.edu	Heather	Hallenbeck
Muzi.Liu@marist.edu	Muzi	Liu
Carol.Friedman@marist.edu	Carol	Friedman
Tony.Carrizales@marist.edu	Tony	Carrizales
Mohammadali.KoorankBeheshti@marist.edu	Mohammadali	Koorank Beheshti
Patricio.Morales@marist.edu	Patricio	Morales
Paul.Ciminello@marist.edu	Paul	Ciminello
Donald.Schwartz@marist.edu	Donald	Schwartz
SangKeun.Yoo@marist.edu	Sang Keun	Yoo
DooRi.Chung@marist.edu	DooRi	Chung
Brian.Cronin2@marist.edu	Brian	Cronin
Aimee.Vargas@marist.edu	Aimee	Vargas-Rodriguez
Patrick.Boylan@marist.edu	Patrick	Boylan
David.Tanguay@marist.edu	David	Tanguay
Kelly.Murray1@marist.edu	Kelly	Murray
Corey.Fenstemacher@marist.edu	Corey	Fenstemacher
...		
Marilyn.Lyons@marist.edu	Marilyn	Lyons
Francis.Kelly@marist.edu	Francis	Kelly
Malinda.Behrens@marist.edu	Malinda	Behrens

(438 rows)

```
CREATE TABLE Schools (
    code char(2) PRIMARY KEY,
    name text
);
```

- Strong entity to represent Marist Schools

Functional Dependency: **code** -> name

Schools - Sample Data

code	name
SM	School of Management
RG	Registrar
CO	School Communication Arts
LA	School of Liberal Arts
SI	School of Science
AL	Arts and Letters
CA	Comm Media Arts
CC	School Computer Sci/ Math
SB	School Behavioral/Social Sci
MG	Management Studies
BY	MA Ed. Psychology
AT	MS Athletic Training
SC	Science
PP	Professional Programs
GP	Global and Profess Programs
CF	MS Global Fashion Merch
BH	Mental Health Counseling
MB	MBA
PA	Physicians Assistant Studies
CX	MA Communication
CS	MSCS Computer Science
CW	Art and Art History
HU	Humanities
BE	MS Education
MP	MPA
BX	MA Psychology
BZ	MA School Psychology
MT	MS Technology Management
BT	MA in Teaching
PT	Dr. Physical Therapy
MA	Professional Accountancy
BS	Behavioral/Social Science

(32 rows)

```sql
CREATE TABLE Subjects (
    code char(4) PRIMARY KEY,
    name text,
    schoolCode char(2) REFERENCES Schools(code)
);
```

- Weak entity to represent Marist Subjects

Functional Dependency: **code** -> name, schoolCode

Subjects – Sample Data

code	name	schoolcode
ACCT	Accounting	SM
ADM	Admissions Intern	RG
ADVT	Advertising	CO
AFRK	Africa	LA
AGR	Agriculture-LdM	SI
AMST	American Studies	LA
ANTH	Anthropology	SI
IANT	Anthropology-International	SI
ARAB	Arabic	LA
ARCH	Architecture - LDM	CO
ART	Art	CO
ARTL	Art History - LDM	CO
STUD	Art Studio - LDM	CO
IART	Art-International	CO
CSNL	Civilization of Netherlands	LA
COM	Communication	CO
CLDM	Communications - LDM	CO
ICOM	Communications-International	CA
CMSC	Computer Science	CC
CSIS	Computer Science Info Sys	CC
CMPT	Computing Technology	CC
...		
SOC	Sociology	SB
ISOC	Sociology-International	SB
SPAN	Spanish	LA

(172 rows)

- Weak entity to represent Marist Courses

```
CREATE TABLE Courses (
    number char(10),
    subjectCode char(4) REFERENCES Subjects(code),
    bannerId text UNIQUE,-- Never trust others
    name text,
    PRIMARY KEY(number, subjectCode)
);
```

- bannerId is not chosen as the primary key, yet has a UNIQUE constraint... this is because if Banner is no longer used or for some reason became inconsistent it would be a much harder migration if it was used for as a PK everywhere. It is still kept as it could be useful for various procedures to have this identifier while we know it is consistent .

Functional Dependency:
number, subjectCode -> bannerId, name

Professor - Sample Data

```
   number   | subjectcode | bannerid |                    name
------------+-------------+----------+-----------------------------------------------
 101N       | ACCT        | 5428     | PRIN OF ACCT I
 102N       | ACCT        | 5429     | PRIN OF ACCT II
 201N       | ACCT        | 5430     | PRIN OF ACCT I
 202N       | ACCT        | 5432     | PRIN ACCTING II
 203N       | ACCT        | 5434     | FINANCIAL ACCTNG
 204N       | ACCT        | 5436     | MANAGERIAL ACCTNG
 301N       | ACCT        | 5438     | INTERM ACCTING I
 302N       | ACCT        | 5440     | INTERM ACCTING II
 303N       | ACCT        | 5442     | ACCT THEORY PRAC
 310N       | ACCT        | 5444     | COST ACCTING
 311N       | ACCT        | 5446     | INFO FOR DECISION
 315N       | ACCT        | 12415    | FRAUD EXAMINATION
 320N       | ACCT        | 5448     | INTERNATIONAL ACCT
 330N       | ACCT        | 15864    | FIN STATMNT ANALYS
 350N       | ACCT        | 5451     | ACCTING SYSTEMS
 380N       | ACCT        | 5453     | ACCOUNT INTERN
 392N       | ACCT        | 5454     | SP TOP ACCOUNTING
 393N       | ACCT        | 5456     | SP TOP ACCOUNTING
 394N       | ACCT        | 5458     | ACCOUNTING INTERN
 395N       | ACCT        | 5460     | ACCOUNT INTERN
 396N       | ACCT        | 5462     | ACCTING INTERN
 397N       | ACCT        | 5464     | ACCOUNT INTERN
 398N       | ACCT        | 5466     | ACCOUNT INTERN
 399N       | ACCT        | 5468     | ACCOUNT INTERN
 ...
 352L       | ISPA        | 10956    | RENEWABLE ENERGIES - SPAIN
 392L       | ISPA        | 15591    | ST:ISPAN
(6044 rows)
```

Marist boasts over 6000 courses!

(many not available every term)

```
CREATE TABLE Sections (
    courseNumber char(10),
    subjectCode char(4),
    number char(4),
    -- term must be a season followed by 4 numbers
    -- note season year is an atomic term for this use case
    term text CHECK (term ~ '^(Winter|Spring|Summer|Fall) \d{4}$'),
    enrollment int CHECK (enrollment >= 0),
    maximumEnrollment int CHECK (enrollment >= 0),
    bannerId text UNIQUE, -- Never trust others
    -- only storing doing email for simplicity of adding data
    primaryProfessor text REFERENCES Professors(email),
    FOREIGN KEY (courseNumber, subjectCode) REFERENCES Courses(number, subjectCode),
    PRIMARY KEY(courseNumber, subjectCode, number, term)
);
```

- Weak entity to represent Marist sections
- BannerId has the same reasoning as with courses
- Term could arguably be brought out to another table especially if other attributes such as whether it is historical or not is needed

Functional Dependency:
courseNumber, subjectCode, number, term-> enrollment, maximumEnrollment, bannerId, primaryProfessor

Sections – Sample Data (1st scrape)

coursenumber	subjectcode	number	term	enrollment	maximumenrollment	bannerid	primaryprofessor
497N	CMPT	111	Fall 2024	0	0	176016	Carolyn.Matheus@marist.edu
498N	CMPT	111	Fall 2024	0	0	176013	Carolyn.Matheus@marist.edu
180N	CONV	190	Fall 2024	0	0	175185	
305N	CONV	180	Fall 2024	0	0	174926	
400L	CONV	180	Fall 2024	0	0	175422	
401L	CONV	180	Fall 2024	0	0	175050	
101L	CRJU	111	Fall 2024	0	24	175508	Samantha.Sayegh@marist.edu
101L	CRJU	112	Fall 2024	0	24	174866	Amanda.Bergold@marist.edu
101L	CRJU	113	Fall 2024	0	24	174419	Amanda.Bergold@marist.edu
101L	CRJU	200	Fall 2024	0	24	176277	Samantha.Sayegh@marist.edu
101L	CRJU	721	Fall 2024	0	24	175720	Sarah.DeLucca1@marist.edu
202L	CRJU	111	Fall 2024	0	24	174091	
202L	CRJU	112	Fall 2024	0	24	175106	
203L	CRJU	111	Fall 2024	0	20	176366	
206N	CRJU	200	Fall 2024	0	24	174092	Edward.Thomas.McLoughlin@marist.edu
230L	CRJU	800	Fall 2024	0	24	175251	Frank.Merenda@marist.edu
230L	CRJU	801	Fall 2024	0	24	174093	Frank.Merenda@marist.edu
235L	CRJU	800	Fall 2024	0	24	174979	Vanessa.Lynn@marist.edu
235L	CRJU	801	Fall 2024	0	24	174980	Vanessa.Lynn@marist.edu
290L	CRJU	190	Fall 2024	0	0	175600	
301N	CRJU	200	Fall 2024	0	25	176900	Steven.Minard@marist.edu
302L	CRJU	111	Fall 2024	0	24	175275	Edward.Thomas.McLoughlin@marist.edu
302L	CRJU	200	Fall 2024	0	24	175276	William.Sayegh@marist.edu
306L	CRJU	111	Fall 2024	0	20	176897	Samantha.Sayegh@marist.edu
306L	CRJU	112	Fall 2024	0	20	175688	Samantha.Sayegh@marist.edu
...							
103L	ECON	114	Fall 2024	0	10	174221	Kole.Camaj@marist.edu
103L	ECON	115	Fall 2024	0	10	177160	Gary.Jacobi@marist.edu
103L	ECON	116	Fall 2024	0	10	174224	Kole.Camaj@marist.edu

(2259 rows)

- Strong entity to represent Marist students

```
CREATE TABLE Students (
    id SERIAL PRIMARY KEY,
    firstName text
);
```

Functional Dependency: **id** -> firstName

Students - Sample Data

```
 id | firstname
----+-----------
  1 | Patrick
  2 | Daisy
  3 | Alan Jr
  4 | Jimmy
(4 rows)
```

```
CREATE TABLE PreferredEnrollments (
    courseNumber char(10),
    subjectCode char(4),
    sectionNumber char(4),
    term text,
    studentId int REFERENCES Students(id),
    FOREIGN KEY (courseNumber, subjectCode, sectionNumber, term)
        REFERENCES Sections(courseNumber, subjectCode, number, term) ON DELETE
CASCADE,
    PRIMARY KEY(courseNumber, subjectCode, sectionNumber, term, studentId)
);
```

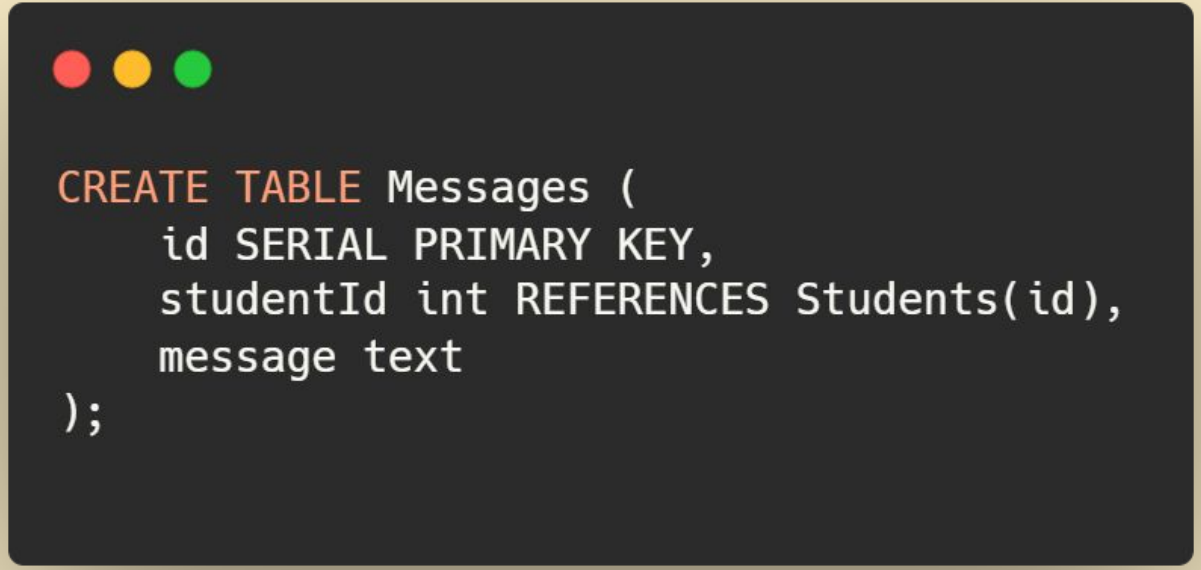
- Weak entity to represent Marist students' preferred enrollments

Functional Dependency: every field is part of pk

Preferred Enrollments – Sample Data

coursenumber	subjectcode	sectionnumber	term	studentid
333N	CMPT	111	Fall 2024	1
422N	CMPT	111	Fall 2024	1
440L	CMPT	111	Fall 2024	1
475N	CMPT	113	Fall 2024	1
476N	CMPT	721	Fall 2024	1
477L	ENG	111	Fall 2024	2
101L	FREN	111	Fall 2024	2
120L	MDIA	111	Fall 2024	2
101L	ART	113	Fall 2024	2
150L	ENG	115	Fall 2024	3
402L	BUS	200	Fall 2024	3
132N	PHED	111	Fall 2024	3
324L	COM	200	Fall 2024	3

(13 rows)



```
CREATE TABLE Messages (
    id SERIAL PRIMARY KEY,
    studentId int REFERENCES Students(id),
    message text
);
```

- Strong entity to represent Marist students' messages
- Not reliant on anything such as sections or meetings to make messages more flexible and resilient to deletions

Functional Dependency: **id** -> studentId, message

Messages – Sample Data
(Generated as several scraped points were added)

```
 id | studentid |                    message
----+-----------+-----------------------------------------------------
  1 |         2 | The following items changed for a meeting of MDIA 120L
                  111 in your enrolled: start-time day duration
  2 |         2 | The following items changed for a meeting of MDIA 120L
                  111 in your enrolled: start-time duration
  3 |         1 | CMPT 475N 113 only has 0 seats left!
  4 |         1 | CMPT 476N 721 only has 0 seats left!
  5 |         3 | The following items changed for a meeting of PHED 132N
                  111 in your enrolled: day
  6 |         3 | The following items changed for a meeting of PHED 132N
                  111 in your enrolled: day
  7 |         2 | The following items changed for a meeting of FREN 101L
                  111 in your enrolled: day
  8 |         2 | The following items changed for a meeting of FREN 101L
                  111 in your enrolled: day
  9 |         3 | COM 324L 200 only has 0 seats left!
 10 |         2 | ART 101L 113 only has 0 seats left!
 11 |         1 | CMPT 422N 111 only has 6 seats left!
 12 |         3 | PHED 132N 111 only has 3 seats left!
(12 rows)
```

```
CREATE TABLE Meetings (
    courseNumber char(4),
    subjectCode char(4),
    sectionNumber char(4),
    term text,
    startTime TIME,
    day dayOfWeek,
    duration INTERVAL,
    FOREIGN KEY (courseNumber, subjectCode, sectionNumber, term)
      REFERENCES Sections(courseNumber, subjectCode, number, term) ON DELETE CASCADE,
    PRIMARY KEY(courseNumber, subjectCode, sectionNumber, term, startTime, day)
);
```

- Weak entity to represent Marist sections' meetings

Functional Dependency:
courseNumber, subjectCode, sectionNumber, term, startTime, day -> duration

Meetings – Sample Data

coursenumber	subjectcode	sectionnumber	term	starttime	day	duration
101L	ITAL	111	Fall 2024	09:30:00	Tuesday	01:15:00
101L	ITAL	111	Fall 2024	09:30:00	Friday	01:15:00
101L	ITAL	112	Fall 2024	11:00:00	Tuesday	01:15:00
101L	ITAL	112	Fall 2024	11:00:00	Thursday	01:15:00
102L	ITAL	111	Fall 2024	09:30:00	Tuesday	01:15:00
102L	ITAL	111	Fall 2024	09:30:00	Friday	01:15:00
105L	ITAL	111	Fall 2024	12:30:00	Tuesday	01:15:00
105L	ITAL	111	Fall 2024	12:30:00	Friday	01:15:00
201L	ITAL	111	Fall 2024	14:00:00	Tuesday	01:15:00
201L	ITAL	111	Fall 2024	14:00:00	Friday	01:15:00
307L	ITAL	111	Fall 2024	15:30:00	Tuesday	01:15:00
307L	ITAL	111	Fall 2024	15:30:00	Thursday	01:15:00
393L	ITAL	111	Fall 2024	14:00:00	Tuesday	01:15:00
393L	ITAL	111	Fall 2024	14:00:00	Friday	01:15:00
101L	JPN	111	Fall 2024	12:30:00	Tuesday	01:15:00
101L	JPN	111	Fall 2024	12:30:00	Friday	01:15:00
105L	JPN	111	Fall 2024	14:00:00	Tuesday	01:15:00
105L	JPN	111	Fall 2024	14:00:00	Friday	01:15:00
192L	LANG	111	Fall 2024	17:00:00	Tuesday	01:15:00
192L	LANG	111	Fall 2024	17:00:00	Thursday	01:15:00
101L	LAT	111	Fall 2024	14:00:00	Tuesday	01:15:00
101L	LAT	111	Fall 2024	14:00:00	Friday	01:15:00
104L	LERN	111	Fall 2024	12:30:00	Tuesday	01:15:00
104L	LERN	111	Fall 2024	12:30:00	Friday	01:15:00
...						
201L	PHYS	113	Fall 2024	14:00:00	Monday	01:15:00
201L	PHYS	113	Fall 2024	14:00:00	Thursday	01:15:00
213L	PHYS	113	Fall 2024	11:00:00	Wednesday	02:45:00

(2270 rows)

View - School to Course

```sql
CREATE OR REPLACE VIEW schoolCourse AS
SELECT sc.code as schoolCode, sc.name as schoolName, su.code as subjectCode, su.name as
subjectName,
    c.name as courseName, c.number as courseNumber
FROM Schools sc
INNER JOIN Subjects su ON sc.code = su.schoolCode
INNER JOIN Courses c ON su.code = c.subjectCode;
```

- Allows easy access for all data typically associated with a course

View Output – School to Course

schoolcode	schoolname	subjectcode	subjectname	coursename	coursenumber
SM	School of Management	ACCT	Accounting	PRIN OF ACCT I	101N
SM	School of Management	ACCT	Accounting	PRIN OF ACCT II	102N
SM	School of Management	ACCT	Accounting	PRIN OF ACCT I	201N
SM	School of Management	ACCT	Accounting	PRIN ACCTING II	202N
SM	School of Management	ACCT	Accounting	FINANCIAL ACCTNG	203N
SM	School of Management	ACCT	Accounting	MANAGERIAL ACCTNG	204N
SM	School of Management	ACCT	Accounting	INTERM ACCTING I	301N
SM	School of Management	ACCT	Accounting	INTERM ACCTING II	302N
SM	School of Management	ACCT	Accounting	ACCT THEORY PRAC	303N
SM	School of Management	ACCT	Accounting	COST ACCTING	310N
SM	School of Management	ACCT	Accounting	INFO FOR DECISION	311N
SM	School of Management	ACCT	Accounting	FRAUD EXAMINATION	315N
SM	School of Management	ACCT	Accounting	INTERNATIONAL ACCT	320N
...					
LA	School of Liberal Arts	ISPA	Spanish-International	HIST OF MARXIST-LEN PHIL II	340L
LA	School of Liberal Arts	ISPA	Spanish-International	RENEWABLE ENERGIES - SPAIN	352L
LA	School of Liberal Arts	ISPA	Spanish-International	ST:ISPAN	392L

(6044 rows)

View - Course to Meeting

```
CREATE OR REPLACE VIEW courseMeeting AS
SELECT c.name as courseName, m.courseNumber as courseNumber, m.subjectCode as subjectCode,
    m.sectionNumber as sectionNumber, s.primaryProfessor as primaryProfessor, m.startTime as startTime,
    m.day as day, m.duration as duration, m.term as term
FROM Courses c
INNER JOIN Sections s ON
   c.number = s.courseNumber AND
   c.subjectCode = s.subjectCode
INNER JOIN Meetings m ON
   m.courseNumber = s.courseNumber AND
   m.subjectCode = s.subjectCode AND
   m.sectionNumber = s.number AND
   m.term = s.term
ORDER BY m.term, m.courseNumber, m.subjectCode, m.sectionNumber, m.day
;
```

- Allows easy access for all data typically associated with a meeting

View Output – Course to Meeting

coursename	coursenumber	subjectcode	sectionnumber	primaryprofessor
FASHION IN CULTURE+COMMERCE	100L	FASH	111	Sonia.Roy@marist.edu
FASHION IN CULTURE+COMMERCE	100L	FASH	111	Sonia.Roy@marist.edu
FASHION IN CULTURE+COMMERCE	100L	FASH	112	Ana.Ortega-Johnson@marist.edu
FASHION IN CULTURE+COMMERCE	100L	FASH	112	Ana.Ortega-Johnson@marist.edu
FASHION IN CULTURE+COMMERCE	100L	FASH	113	Ana.Ortega-Johnson@marist.edu
FASHION IN CULTURE+COMMERCE	100L	FASH	113	Ana.Ortega-Johnson@marist.edu
FASHION IN CULTURE+COMMERCE	100L	FASH	114	Stephanie.Conover@marist.edu
FASHION IN CULTURE+COMMERCE	100L	FASH	114	Stephanie.Conover@marist.edu
FASHION IN CULTURE+COMMERCE	100L	FASH	115	Sonia.Roy@marist.edu
FASHION IN CULTURE+COMMERCE	100L	FASH	115	Sonia.Roy@marist.edu
FASHION IN CULTURE+COMMERCE	100L	FASH	116	Anthony.Millero@marist.edu
FASHION IN CULTURE+COMMERCE	100L	FASH	116	Anthony.Millero@marist.edu
ELEM ARAB I	101L	ARAB	111	Al.Abdelrahman@marist.edu
ELEM ARAB I	101L	ARAB	111	Al.Abdelrahman@marist.edu
FUND-ART+DESIGN	101L	ART	111	Fletcher.Boote@marist.edu
FUND-ART+DESIGN	101L	ART	111	Fletcher.Boote@marist.edu
FUND-ART+DESIGN	101L	ART	112	Fletcher.Boote@marist.edu
FUND-ART+DESIGN	101L	ART	112	Fletcher.Boote@marist.edu
FUND-ART+DESIGN	101L	ART	113	Elisa.Lendvay@marist.edu
FUND-ART+DESIGN	101L	ART	113	Elisa.Lendvay@marist.edu
FUND-ART+DESIGN	101L	ART	114	Laura.Hammond-Toonkel@marist.edu
FUND-ART+DESIGN	101L	ART	114	Laura.Hammond-Toonkel@marist.edu
FUND-ART+DESIGN	101L	ART	200	Julia.Barnes@marist.edu
FUND-ART+DESIGN	101L	ART	201	Steven.Petruccio@marist.edu
...				
INFO SYST PROJ	720L	MSIS	232	Brian.Gormanly@marist.edu
INFO SYS POLICY	730L	MSIS	232	Dennis.Rush@marist.edu
INFO SYS POLICY	730L	MSIS	232	Dennis.Rush@marist.edu

(2270 rows)

starttime	day	duration	term
12:30:00	Monday	01:15:00	Fall 2024
12:30:00	Thursday	01:15:00	Fall 2024
15:30:00	Monday	01:15:00	Fall 2024
15:30:00	Wednesday	01:15:00	Fall 2024
17:00:00	Monday	01:15:00	Fall 2024
17:00:00	Wednesday	01:15:00	Fall 2024
12:30:00	Tuesday	01:15:00	Fall 2024
12:30:00	Friday	01:15:00	Fall 2024
14:00:00	Monday	01:15:00	Fall 2024
14:00:00	Thursday	01:15:00	Fall 2024
11:00:00	Tuesday	01:15:00	Fall 2024
11:00:00	Thursday	01:15:00	Fall 2024
15:30:00	Monday	01:15:00	Fall 2024
15:30:00	Wednesday	01:15:00	Fall 2024
09:30:00	Monday	01:15:00	Fall 2024
09:30:00	Thursday	02:45:00	Fall 2024
11:00:00	Monday	02:45:00	Fall 2024
12:30:00	Thursday	01:15:00	Fall 2024
15:30:00	Monday	01:15:00	Fall 2024
14:00:00	Wednesday	02:45:00	Fall 2024
14:00:00	Monday	01:15:00	Fall 2024
14:00:00	Thursday	02:45:00	Fall 2024
17:00:00	Tuesday	03:45:00	Fall 2024
18:30:00	Thursday	03:45:00	Fall 2024
17:00:00	Wednesday	01:15:00	Fall 2024
17:00:00	Tuesday	01:15:00	Fall 2024
17:00:00	Thursday	01:15:00	Fall 2024

Report 1

- Get all meeting information for meetings where the course's primary professor starts with 'Al'
- Many marist students enjoy taking classes taught by 'Al' prefixed professor first names

```
SELECT p.firstName, cm.courseName,
cm.sectionNumber, cm.day, cm.startTime,
cm.duration
FROM courseMeeting cm
INNER JOIN Professors p ON p.email =
cm.primaryProfessor
WHERE p.firstName ILIKE 'Al%';
```

Report 1 output

firstname	coursename	sectionnumber	day	starttime	duration
Allaeddin	ELEM ARAB I	111	Monday	15:30:00	01:15:00
Allaeddin	ELEM ARAB I	111	Wednesday	15:30:00	01:15:00
Alquan	PUBLIC PRESENTAT.	112	Monday	08:00:00	01:15:00
Alquan	PUBLIC PRESENTAT.	112	Thursday	08:00:00	01:15:00
Alexander	TECHNOLOGY FOR 21st CENTURY	112	Tuesday	09:30:00	01:15:00
Alexander	TECHNOLOGY FOR 21st CENTURY	112	Friday	09:30:00	01:15:00
Allaeddin	INTERMEDIATE ARABIC I	111	Monday	17:00:00	01:15:00
Allaeddin	INTERMEDIATE ARABIC I	111	Wednesday	17:00:00	01:15:00
Alyssa	SPORTS IN SOCIETY	111	Tuesday	11:00:00	01:15:00

Report 2

- Get identifying section information for all CMPT courses which have meetings but not at 8:00 or on Friday

- Many marist students have contempt for 8:00am's and Friday classes

```sql
SELECT c.name as courseName, s.number as sectionNumber
FROM Courses c
INNER JOIN Sections s ON
   c.number = s.courseNumber AND
   c.subjectCode = s.subjectCode
WHERE
    c.subjectCode='CMPT' AND
    (c.number, c.subjectCode, s.term, s.number) NOT IN (
        SELECT m.courseNumber, m.subjectCode, m.term, m.sectionNumber
        FROM Meetings m
        WHERE
        startTime = '08:00:00' OR
        day = 'Friday'
    )
    -- must have some meetings
    AND (c.number, c.subjectCode, s.term, s.number) IN (
        SELECT m.courseNumber, m.subjectCode, m.term, m.sectionNumber FROM Meetings m
    )
;
```

Report 2 output

coursename	sectionnumber
TECHNOLOGY FOR 21st CENTURY	115
TECHNOLOGY FOR 21st CENTURY	116
MS EXCEL	200
MS EXCEL	201
INTRO TO PROGRAMMING	200
INTRO TO PROGRAMMING	111
INTRO TO PROGRAMMING	115
INTRO TO PROGRAMMING	114
SOFTWARE DEVELOPMENT II	112
SOFTWARE DEVELOPMENT II	201
SOFTWARE DEVELOPMENT II	111
SOFTWARE DEVELOPMENT II	200
TECNOLOGY,ETHICS,+SOCIETY	111
TECNOLOGY,ETHICS,+SOCIETY	112
DATA COMMUNICATIONS	114
DATA COMMUNICATIONS	113
DATA COMMUNICATIONS	111
DATA COMMUNICATIONS	112
INTERNETWORKING	112
INTERNETWORKING	111
DATABASE MANAGEMENT	200
DATABASE MANAGEMENT	201
DATABASE MANAGEMENT	112
DATABASE MANAGEMENT	111
ARCH-HARDWARE+SYSTEMS SOFTWARE	111
SYSTEM DESIGN	111
LANGUAGE STUDY	111
ARTIFICAL INTELLIGENCE	200
GAME DESIGN+PROG I	111
INTRO TO CYBERSECURITY	111
COMPUTER FORENSICS	111
COMPUTER ORG & ARCH	112
COMPUTER ORG & ARCH	111
ALGORITHM ANALYSIS+DESIGN	111
FORMAL LANGUAGES+COMPUTABILITY	111
CS PROJECT	112
CS PROJECT	113
CS PROJECT	111

(38 rows)

Report 3

- Get course information on preferred enrollment sections of student with first name ‘Alan Jr’

- Marist students need all Course information to submit prereq and capacity override forms and discuss courses with advisors

```
SELECT sc.schoolCode, sc.schoolName, sc.subjectCode, sc.subjectName, sc.courseName
FROM schoolCourse sc
INNER JOIN Sections se ON
    sc.courseNumber = se.courseNumber AND
    sc.subjectCode = se.subjectCode
INNER JOIN PreferredEnrollments p ON
    sc.courseNumber = p.courseNumber AND
    se.number = p.sectionNumber AND
    sc.subjectCode = p.subjectCode AND
    se.term = p.term
INNER JOIN Students st ON st.id = p.studentId
WHERE st.firstName = 'Alan Jr'
;
```

*probably better suited as a procedure with input as student but because it is for Alan Jr the system has this hardcoded

Report 3 output

```
 schoolcode |         schoolname          | subjectcode |    subjectname     |        coursename
------------+-----------------------------+-------------+--------------------+---------------------------
 LA         | School of Liberal Arts      | ENG         | English            | INTRO TO THEATRE
 SM         | School of Management        | BUS         | Business           | LABOR RELATIONS
 SI         | School of Science           | PHED        | Physical Education | BOXING
 CO         | School Communication  Arts | COM         | Communication      | RESEARCH+CONSUMER INSIGHT
(4 rows)
```

Report 4

- Get the count of all sections by school name
- Registrar may find this breakdown interesting to see

```
SELECT schoolName, Count(*) as sectionCount
FROM schoolCourse
GROUP BY schoolName;
```

```
          schoolname          | sectioncount
------------------------------+--------------
 Mental Health Counseling     |           28
 Global and Profess Programs  |            6
 Humanities                   |            2
 School Computer Sci/ Math    |          458
 MA Psychology                |           65
 Professional Accountancy     |            7
 MS Education                 |           16
 Dr. Physical Therapy         |           42
 MS Global Fashion Merch      |           12
 School Behavioral/Social Sci |          429
 School Communication  Arts   |         1133
 MSCS Computer Science        |          122
 MA Ed. Psychology            |           65
 School of Science            |          673
 Comm Media Arts              |           70
 MA School Psychology         |           34
 School of Management         |          335
 Registrar                    |          382
 MS Technology Management     |           14
 MA Communication             |           30
 MA in Teaching               |           13
 Physicians Assistant Studies |           31
 Art and Art History          |           24
 MPA                          |           49
 MBA                          |           94
 Science                      |            1
 School of Liberal Arts       |         1806
 Professional Programs        |          103
(28 row)
```

```sql
CREATE OR REPLACE FUNCTION upsert_sections_in_term(
    term_text text,
    banner_ids text[],
    course_numbers text[],
    enrollments int[],
    maximum_enrollments int[],
    subject_codes text[],
    numbers text[],
    primary_professor_emails text[]
)
RETURNS void AS $$
DECLARE
    i int;
BEGIN
    -- delete sections
    DELETE FROM Sections as s
    WHERE
        s.term like term_text AND
        s.BannerId NOT IN (SELECT unnest(banner_ids));

    FOR i IN 1..array_length(banner_ids, 1) LOOP
        INSERT INTO Sections
            (bannerId, courseNumber, subjectCode, number, enrollment, maximumEnrollment, term,
primaryProfessor)
        VALUES (
            banner_ids[i],
            course_numbers[i],
            subject_codes[i],
            numbers[i],
            enrollments[i],
            maximum_enrollments[i],
            term_text,
            primary_professor_emails[i])
        -- Would mean that either ref errors from coures composite key or
        --   the section already exists which in that case should update
        ON CONFLICT (bannerId) DO UPDATE
        SET courseNumber = course_numbers[i],
            subjectCode = subject_codes[i],
            number = numbers[i],
            enrollment = enrollments[i],
            maximumEnrollment = maximum_enrollments[i],
            term = term_text,
            primaryProfessor = primary_professor_emails[i];
        -- Would fail again on ref errors which is wanted
    END LOOP;
END;
$$ LANGUAGE plpgsql
```

Procedure - Upsert Sections

- Given all the updated sections for a given term harmonize it with the current sections
- If the section is not in the updated section delete it else update it with the new values

https://github.com/Pjt727/CMPT308/blob/main/finalProject/upsertSections.sql

Procedure Example Call – Upsert Sections

```
DO $$
DECLARE
    term text := 'Fall 2024';
    banner_ids text[] := ARRAY['175611','175612','175613', ...];
    course_numbers text[] := ARRAY['401L','401L','401L', ...];
    enrollments int[] := ARRAY[0,0,0, ...];
    maximum_enrollments int[] := ARRAY[0,0,0, ...];
    subject_codes text[] := ARRAY['HONR','HONR','HONR', ...];
    numbers text[] := ARRAY['143','144','145', ...];
    primary_professor_emails text[] := ARRAY[NULL,NULL,NULL, ...];
BEGIN
    PERFORM upsert_sections_in_term(
        term,
        banner_ids,
        course_numbers,
        enrollments,
        maximum_enrollments,
        subject_codes,
        numbers,
        primary_professor_emails
    );
END $$;
```

- The output is the side effect this procedure has on the Sections table by updating all sections in that term
- It is also important for the triggers that this procedure calls the correct create, update, delete SQL action instead of just deleting all and then remaking

```
CREATE OR REPLACE FUNCTION upsert_meetings_in_section(
    course_number text,
    subject_code text,
    section_number text,
    term_text text,
    start_times TIME[],
    days dayOfWeek[],
    durations INTERVAL[]
)
RETURNS void AS $$
DECLARE
    i int;
    do_create_index boolean;
    dont_update boolean;
    meeting_count INTEGER;
    created_indices int[];
    -- work around to update a single record bc you can not limit on an update
    start_time_to_update TIME;
    day_to_update dayOfWeek;
    debugging RECORD;
BEGIN
    SELECT COUNT(*) INTO meeting_count FROM Meetings m
    WHERE
        m.courseNumber = course_number AND
        m.subjectCode = subject_code AND
        m.sectionNumber = section_number AND
        m.term = term_text
    ;

    ...

        END LOOP;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Procedure – Upsert Meetings

- Given all the updated meetings for a given section harmonize it with the current meeting

- Make the least amount of changes (e.i. If one meeting's day changes interpret that as the only update and do not update any other meeting)

FULL SCRIPT NOT SHOWN, SEE: https://github.com/Pjt727/CMPT308/blob/main/finalProject/upsertSections.sql

Procedure Example Call – Upsert Meetings

- Several calls to the upsert meeting procedure in between each showing the all meetings for that section

```
SELECT upsert_meetings_in_section('201L', 'ITAL', '111', 'Fall 2024',
    ARRAY['14:00:00','14:00:00']::TIME[], ARRAY['Tuesday','Monday']::dayOfWeek[],
    ARRAY['75m','75m']::INTERVAL[]);
SELECT * FROM Meetings WHERE courseNumber='201L' AND subjectCode='ITAL';
SELECT upsert_meetings_in_section('201L', 'ITAL', '111', 'Fall 2024',
    ARRAY[]::TIME[], ARRAY[]::dayOfWeek[], ARRAY[]::INTERVAL[]);
SELECT * FROM Meetings WHERE courseNumber='201L' AND subjectCode='ITAL';
SELECT upsert_meetings_in_section('201L', 'ITAL', '111', 'Fall 2024',
    ARRAY['14:00:00','14:00:00']::TIME[], ARRAY['Tuesday','Friday']::dayOfWeek[],
    ARRAY['75m','75m']::INTERVAL[]);
SELECT * FROM Meetings WHERE courseNumber='201L' AND subjectCode='ITAL';
SELECT upsert_meetings_in_section('201L', 'ITAL', '111', 'Fall 2024',
    ARRAY['14:00:00']::TIME[], ARRAY['Thursday']::dayOfWeek[],
    ARRAY['75m','75m']::INTERVAL[]);
SELECT * FROM Meetings WHERE courseNumber='201L' AND subjectCode='ITAL';
```

- The output (on the next pag demonstrates how the meetings change… furthermore the messages table demonstrates the correct SQL action (delete, update, create) is made

Procedure Example Call Output – Upsert Meetings

```
 coursenumber | subjectcode | sectionnumber |   term    | starttime |   day    | duration
--------------+-------------+---------------+-----------+-----------+----------+----------
 201L         | ITAL        | 111           | Fall 2024 | 14:00:00  | Tuesday  | 01:15:00
 201L         | ITAL        | 111           | Fall 2024 | 14:00:00  | Thursday | 01:15:00
(2 rows)

 coursenumber | subjectcode | sectionnumber | term | starttime | day | duration
--------------+-------------+---------------+------+-----------+-----+----------
(0 rows)

 coursenumber | subjectcode | sectionnumber |   term    | starttime |   day   | duration
--------------+-------------+---------------+-----------+-----------+---------+----------
 201L         | ITAL        | 111           | Fall 2024 | 14:00:00  | Tuesday | 01:15:00
 201L         | ITAL        | 111           | Fall 2024 | 14:00:00  | Friday  | 01:15:00
(2 rows)

 coursenumber | subjectcode | sectionnumber |   term    | starttime |   day    | duration
--------------+-------------+---------------+-----------+-----------+----------+----------
 201L         | ITAL        | 111           | Fall 2024 | 14:00:00  | Thursday | 01:15:00
(1 row)
```

Trigger – Update Meetings

- Notifies students through the message table when a meeting has changed and what field(s)

- Below is sample output from when various meetings had changed after loading some of the sample data

https://github.com/Pjt727/CMPT308/blob/main/finalProject/meetingTriggers.sql

```
CREATE OR REPLACE FUNCTION meeting_update()
RETURNS TRIGGER AS $$
DECLARE
    students int[];
    student int;
    do_message boolean;
    message text;
BEGIN
    -- The only reason to notify is if the day/ time/ duration changed
    message := 'The following items changed for a meeting of ';
    message := message || NEW.subjectCode || ' ' || NEW.courseNumber || ' ' || NEW.sectionNumber;
    message := message || ' in your enrolled: ';
    do_message := false;

    IF OLD.startTime != NEW.startTime THEN
        do_message := true;
        message := message || 'start-time ';
    END IF;

    IF OLD.day != NEW.day THEN
        do_message := true;
        message := message || 'day ';
    END IF;

    IF OLD.duration != NEW.duration THEN
        do_message := true;
        message := message || 'duration ';
    END IF;

    IF do_message THEN
        RAISE NOTICE '%, %, %', NEW.courseNumber, NEW.subjectCode, NEW.sectionNumber;
        SELECT ARRAY_AGG(p.studentID)
        INTO students
        FROM PreferredEnrollments p
        WHERE
            p.courseNumber = NEW.courseNumber AND
            p.subjectCode = NEW.subjectCode AND
            p.sectionNumber = NEW.sectionNumber AND
            p.term = NEW.term;

        IF array_length(students, 1) >= 1 THEN
            FOREACH student in ARRAY students LOOP
                INSERT INTO Messages (studentId, message)
                VALUES(student, message);
            END LOOP;
        END IF;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER meeting_update
AFTER UPDATE ON Meetings
FOR EACH RO
EXECUTE FUNCTION meeting_update();
```

```
 id | studentid |                                                                                                  message
----+-----------+-----------------------------------------------------------------------------------------------------------------------------------------------------
  1 |         2 | The following items changed for a meeting of MDIA 120L 111 in your enrolled: start-time day duration
  2 |         2 | The following items changed for a meeting of MDIA 120L 111 in your enrolled: start-time duration
  5 |         3 | The following items changed for a meeting of PHED 132N 111 in your enrolled: day
  6 |         3 | The following items changed for a meeting of PHED 132N 111 in your enrolled: day
  7 |         2 | The following items changed for a meeting of FREN 101L 111 in your enrolled: day
  8 |         2 | The following items changed for a meeting of FREN 101L 111 in your enrolled: day
(6 rows)
```

```
CREATE OR REPLACE FUNCTION capacity_notice()
RETURNS TRIGGER AS $$
DECLARE
    students int[];
    student int;
    message text;
    do_message boolean;
BEGIN
    -- do nothing if the enorllments are the same
    IF OLD.enrollment = NEW.enrollment THEN
        RETURN NEW;
    END IF;
    do_message := false;
    -- if the enrollment is within 25% of the capicity then give a message to every student
    --   that has this in there enrollment
    IF NEW.enrollment >= NEW.maximumEnrollment * .75 THEN
        do_message := true;
        message := NEW.subjectCode || ' ' || NEW.courseNumber || ' '
            || NEW.number || ' only has ' || (NEW.maximumEnrollment - NEW.enrollment) || ' seats left!';
    ELSIF OLD.enrollment > NEW.enrollment THEN
        do_message := true;
        message := NEW.subjectCode || ' ' || NEW.courseNumber || ' '
            || NEW.number || ' has ' || (OLD.enrollment - NEW.enrollment) || ' new seats and ' ||
            (NEW.maximumEnrollment - NEW.enrollment) || ' seats left!!';
    END IF;

    IF do_message THEN
        RAISE NOTICE '%, %, %, %', NEW.courseNumber, new.subjectCode, new.number, new.term;
        RAISE NOTICE '%', message;
    END IF;

    IF do_message THEN
        SELECT p.studentID
        INTO students
        FROM PreferredEnrollments p
        WHERE
            p.courseNumber = NEW.courseNumber AND
            p.subjectCode = NEW.subjectCode AND
            p.sectionNumber = NEW.number AND
            p.term = NEW.term;
        IF students != NULL THEN
            FOREACH student in ARRAY students LOOP
                INSERT INTO Messages (studentId, message)
                VALUES(student, message);
            END LOOP;
        END IF;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER capacity_notice
AFTER UPDATE ON SECTIONS
FOR EACH RO
EXECUTE FUNCTION capacity_notice();
```

Trigger – Update Meetings

- Notifies students through the message table when a section is filling up or has gotten more available seats

- Below is sample output from when various sections had changed after loading some of the sample data

https://github.com/Pjt727/CMPT308/blob/main/finalProject/capacityTrigger.sql

```
 id | studentid |                 message
----+-----------+-----------------------------------------
  3 |         1 | CMPT 475N 113 only has 0 seats left!
  4 |         1 | CMPT 476N 721 only has 0 seats left!
  9 |         3 | COM 324L 200 only has 0 seats left!
 10 |         2 | ART 101L 113 only has 0 seats left!
 11 |         1 | CMPT 422N 111 only has 6 seats left!
 12 |         3 | PHED 132N 111 only has 3 seats left!
(6 rows)
```

Security Notes

- Teachers only have view access on all tables as their only use case would be advising

- Students (or the api students interact through) have view access on all tables to complete their schedule, update access on students to change their name, and update, insert, delete access on their preferred enrollments and messages to manage their information (example use cases: add preferred sections and delete read messages)

```
CREATE ROLE teacher;
GRANT SELECT ON all tables in schema public TO student;

CREATE ROLE student;
GRANT SELECT ON all tables in schema public TO student;
GRANT UPDATE ON Students TO student;
GRANT UPDATE, INSERT, DELETE ON PreferredEnrollments, Messages TO student;

CREATE ROLE registrar;
GRANT ALL ON all tables in schema public TO registrar;
```

- The registrar has access to all powers on each table as it must manage all classes and student data (example use cases: add/ removing students, messages, and classes)

Implementation Notes

The accuracy of section upserts relies on the consistency of Banner id's which are not expected to be directly controlled by the system and thus section upserts must be closely monitored

The database component of the system relies on a separate piece to retrieve the data from Banner and parse it into insert statements or procedure calls

Data ran through procedures is assumed to be in full and problems will occur if the scraping part of the system does not obtain data for a term/ section in its entirety

Known Problems

Students may prefer sections which overlap in time

Students may want to get notified on section additions for certain courses or changes of sections that are not preferred enrollment

Certain procedures are very inefficient especially as the database scales (IN and NOT IN are very expensive operations and are used liberally)

Future Enhancements

Add procedures / triggers to notify students on timing conflicts on meeting changes

Allow students to get notified on section additions/ changings on section courses

Optimize queries perhaps putting more weight on non-database parts of the system especially the differentiating step of meetings and sections (the stored procedures for upsert meetings/sections were painful to write and will be difficult to maintain)

Have simple procedure interfaces for determining sections that overlap in time

Keep logs of certain data points to create trends on items such as enrollments graphs for individual courses